Harmful Algae Bloom Detection System

Team: Gradient Descents

Overview

Problem

- HABs cause $82M+ annual loss
- Manual detection is slow
- Poor coverage, slow response

Objective

- Predict HABs with 90% accuracy
- Automate satellite monitoring
- Provide 8-day forecasts

Target Users

- Environmental agencies (NOAA)
- Coastal industries (Fishing)
- Marine researchers

Why It's Needed

- 5-7 days needed for action
- Current: reactive response
- Proactive prevention needed
- Risk to economy + health

Challenge Area

- Environmental monitoring
- Prediction

Overview

Current Solutions vs Our System

Manual Sampling
- Lab-based analysis
- Days to process
- <1% area coverage
- Reactive detection

NOAA HAB-OFS
- 3-day forecast
- ~60% accuracy
- Florida only

Our HABNet System
- 8-day prediction
- ~90% accuracy
- Globally scalable

Technology Stack

Data sources and Preprocessing

Satellite Data

Earthdata API,
MODIS-Aqua.nc files

Preprocessing

Pandas, Matplotlib,
Xarray, NetCDF4

Label Source and Storage

HABSOS CSV,
Local disk

ML and Model Training

TensorFlow · Keras · PyTorch · NumPy · CO · kaggle Notebooks

Backend

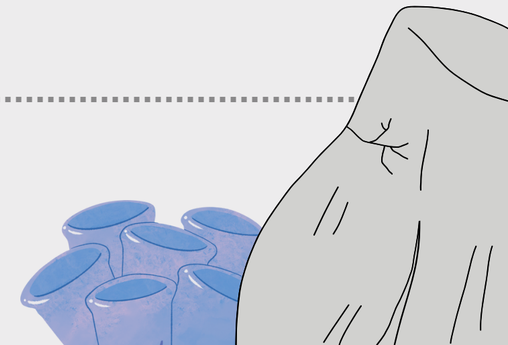
FastAPI · Firebase Firestore

Frontend

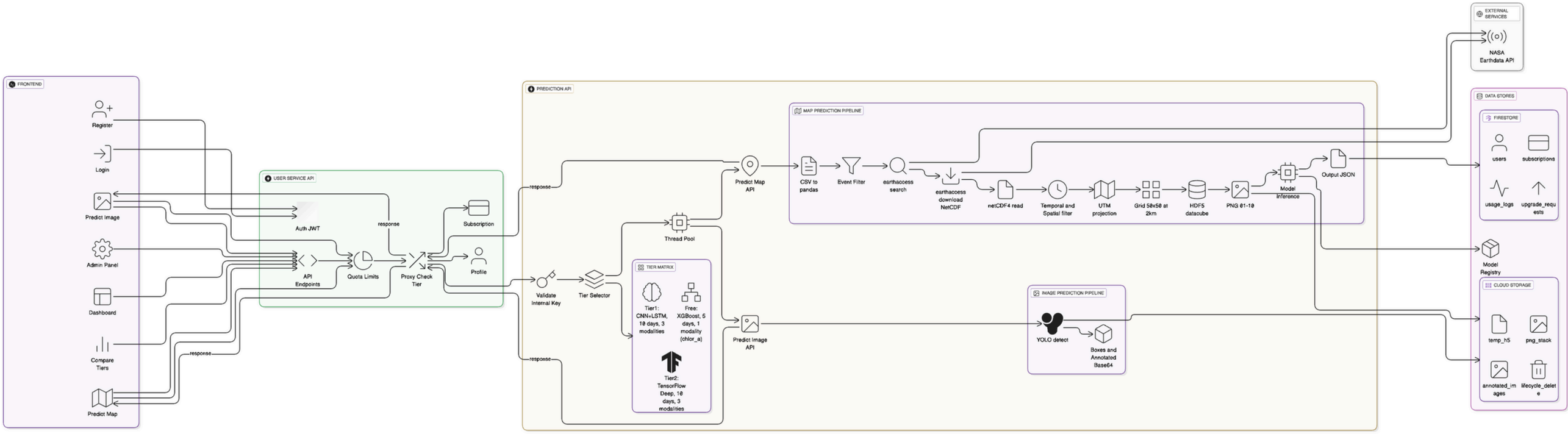
Next.js · Leaflet.js · CSS

DevOps & Deployment

Google Cloud Run · Docker · GitHub · Google Cloud Platform



System Architecture



Data Sources

Ground truth data

HABSOS Database (**NOAA**)

- Karenia brevis cell counts, Lon/Lat coordinates, date
- Over 210k labelled HAB events up to 2018, Gulf of Mexico
- Access: Direct CSV download, processed in our pipeline
- Status: Successfully processed 3,000 high quality events

Data Sources

Satellite data

Access: EarthAccess API with NASA Earthdata auth (1000 req/hr limit)
Available Data:

- Chlorophyll-a concentration (primary indicator)
- Photosynthetically available radiation (PAR)
- Remote sensing reflectance (Rrs bands: 412, 443, 488, 531, 555nm)

Testing Status: Successfully downloading and processing over 3,000 L2 granules (~40MB each)

Data Sources

Datacube data

- 100km × 100km × 10day datacubes around each HAB event
- Extract 7 modalities(data channels) from satellite granules
- Process 10 days leading up to each event
- Discard datacubes with less than 60% data coverage
- 5MB size per datacube
- Storage: Stored in shared drive after processing

Data Sources

Image Generation Pipeline

- **Input**: H5 Datacubes
- Converts 100km×100km×10day datacubes to PNG sequences
- 100×100 pixel standardized images
- Handles UTM coordinate conversion & interpolation
- **Output**: 70 PNG images per datacube (7 modalities × 10 days)
- Successfully processed 3,000+ datacubes

Key Technical Challenges

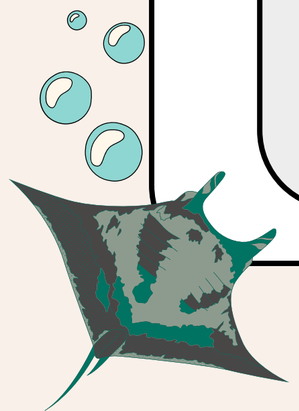
Data Uncertainty

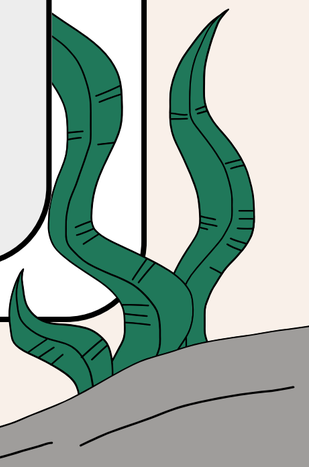
Problem: Satellite data had gaps from cloud cover, noise, and atmospheric effects, making sequences inconsistent.
Solution: Used linear interpolation to fill missing frames, rejected datacubes with less than 60% valid data, and trained on partially incomplete inputs.
Result: Achieved over 90% accuracy even with up to 30% missing data.

Key Technical Challenges

Mass Satellite Data Processing & API Constraints

Problem: Needed to process ~1.2 TB for 3000 HAB events over 10 days, with NASA's API capped at 1000 requests/hour.
Solution: Batched downloads, processed all modalities in parallel, and compressed results to ~5-10 MB per datacube.
Result: Built a high-resolution dataset without exceeding API limits or overloading memory.

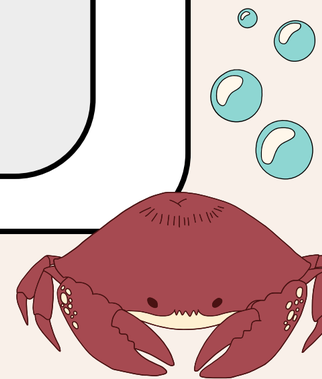




Key Technical Challenges

Slow Predictions & Processing Delays

Problem: Predictions were delayed by slow datacube loading and preprocessing.
Solution: Used multi-threading to process datacubes for multiple days in parallel.
Result: Cut down lag significantly, with final prediction time now mostly dependent on hardware.



Key Technical Challenges

Handling Different User Needs & Resource Costs

Problem: Some users wanted quick map-based predictions, others uploaded images to detect algae's presence, which required heavier models like YOLO or CNN+LSTM.

Solution: Added a tiered system—free tier for lightweight coordinate-based predictions, paid tier for image uploads.

Result: Managed resources effectively, kept response times fast, and avoided unnecessary backend load.

Key Technical Challenges

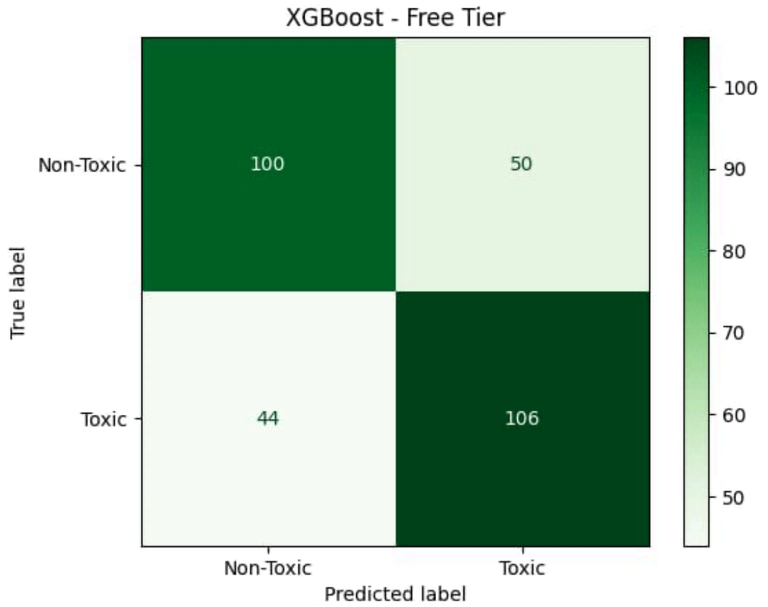
YOLO Object Detection Limitations

Problem: YOLO often detected anything green as algae, and the mixed image resolutions in the training set hurt accuracy.
Solution: We expanded the dataset with image augmentation, which improved precision, but recall stayed below 50%.
Result: Detection improved, but it still confused other green objects with algae. A larger set of consistent, varied images is needed.

Model Evaluation

Model Performance evaluation : **XGBoost**
Base model for **free tier**

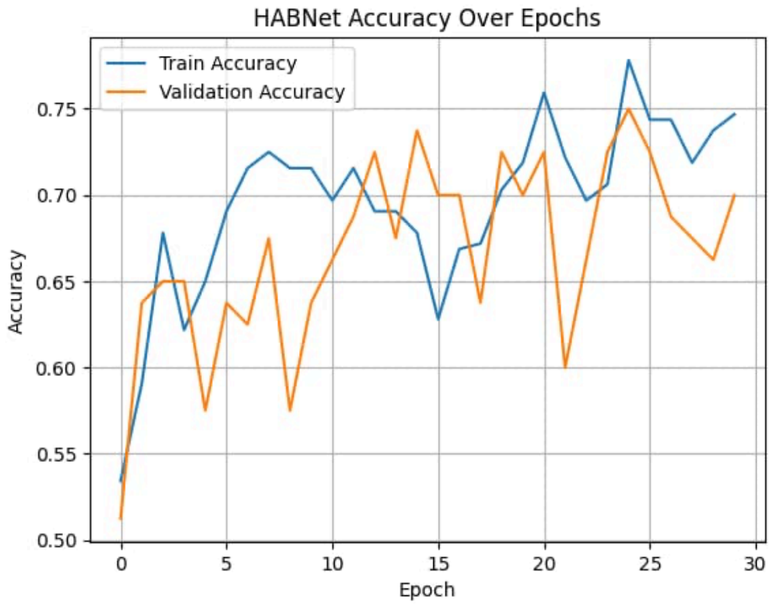
Model	Accuracy
Logistic Regression	92%
Ridge Classifier	90%
XG Boost	**94.2%**
Decision Tree	85%
Random Forest	88%
CNN + LSTM	**88.4%**

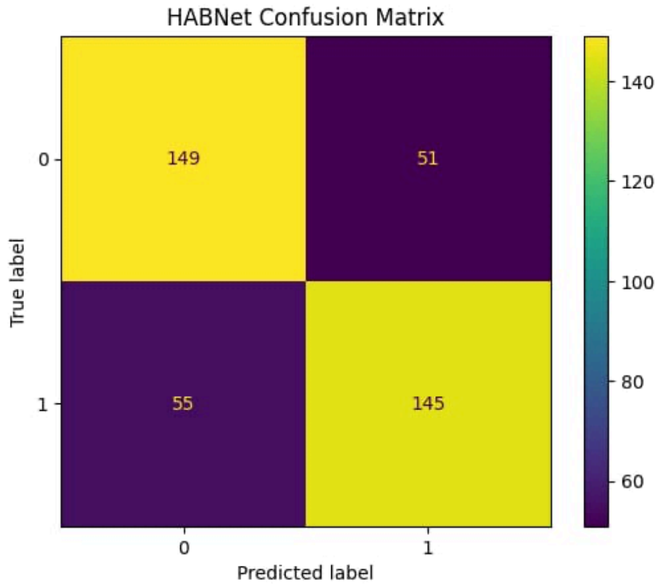


	precision	recall	f1-score	support
0	0.69	0.67	0.68	150
1	0.68	0.71	0.69	150
accuracy			0.69	300
macro avg	0.69	0.69	0.69	300
weighted avg	0.69	0.69	0.69	300

Model Evaluation

Model Performance evaluation : **CNN model**
Base model for **Tier 1**

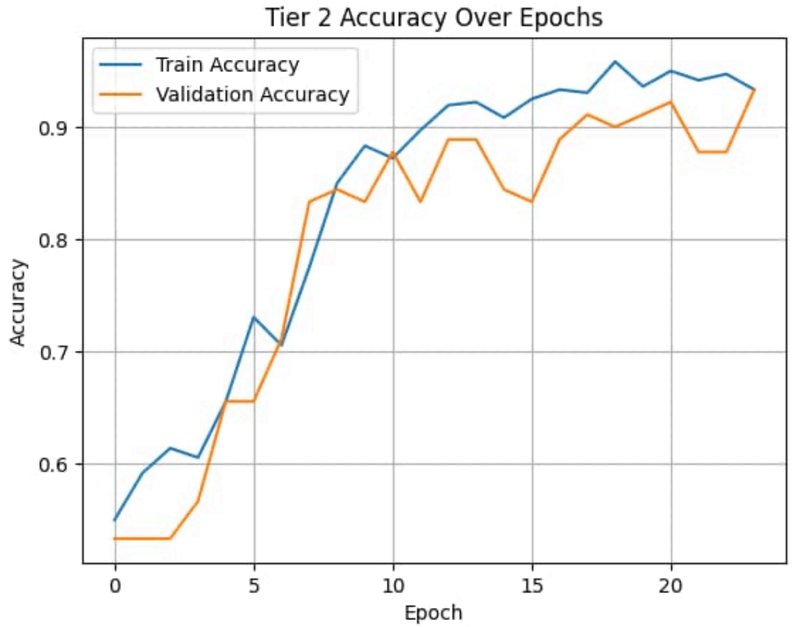


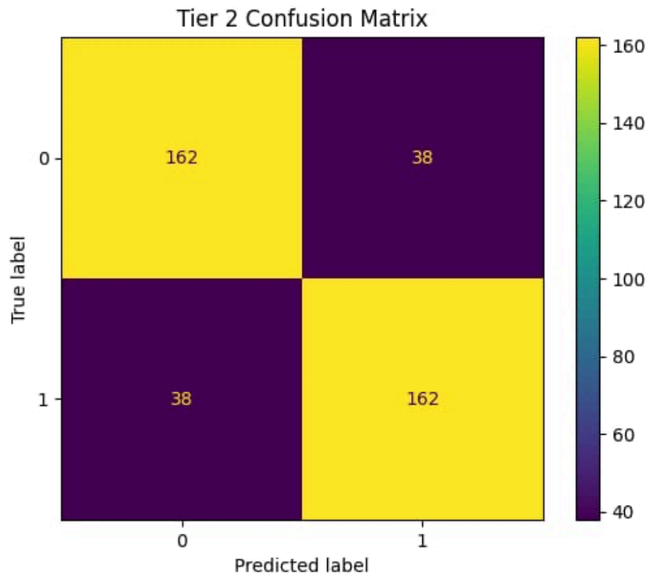


	precision	recall	f1-score	support
0	0.73	0.74	0.74	200
1	0.74	0.72	0.73	200
accuracy			0.73	400
macro avg	0.74	0.73	0.73	400
weighted avg	0.74	0.73	0.73	400

Model Evaluation

Model Performance evaluation : **HabNet like CNN+LSTM model**
Base model for **Tier 2**





metrics for YOLO model

```
{'metrics/precision(B)': 0.8467131524868463,
 'metrics/recall(B)': 0.43670886075949367,
 'metrics/mAP50(B)': 0.4892043112525861,
 'metrics/mAP50-95(B)': 0.2954113329234976,
 'fitness': 0.31479063075640645}
```

	precision	recall	f1-score	support
0	0.81	0.81	0.81	200
1	0.81	0.81	0.81	200
accuracy			0.81	400
macro avg	0.81	0.81	0.81	400
weighted avg	0.81	0.81	0.81	400

User Evaluation

Methodology:

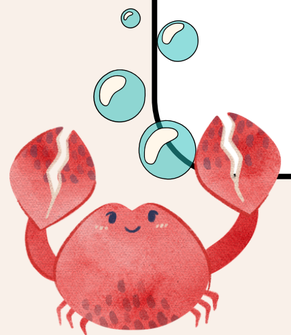
- Survey sent to researchers

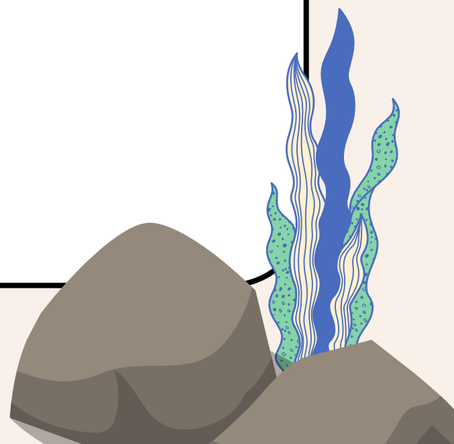
Key Findings:

- Website navigation and UI: intuitive and well-designed
- Tiered subscription model: well-received concept

Areas for Improvement & Actions Planned:

- Need more explanatory content (methodology, data modalities)
- Requested clearer pricing information





Performance Evaluation

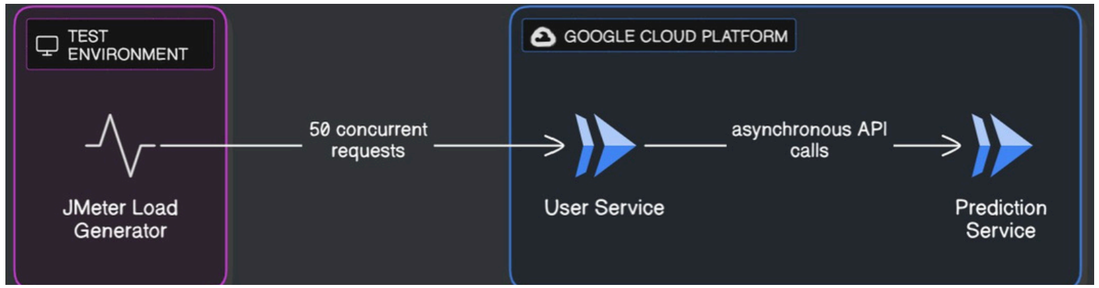
Test Objectives

- **Stability:** Verify system resilience under a 50 user concurrent load.
- **Performance:** Measure key user-facing metrics (response time, throughput).
- **Scalablity:** Obsersve the auto-scaling behaviour of backend cloud run services.

Methodology

- **Tool:** Apache JMeter
- **Load Profile:** 50 concurrent Virtual Users
- **Ramp-up Period:** 5 seconds (10 users/sec)
- **Test Scenario:** Each user makes one request to geospatial prediction endpoint.

System Architecture Under Test



Client-Side Performance Metrics

Key Performance Indicators (User Perspective)

Metric	Result
Success Rate	100% (0.00% Error Rate)
95th Percentile Response Time	75.8 sec
Average Response Time	59.7 seconds
Throughput	0.63 requests/second

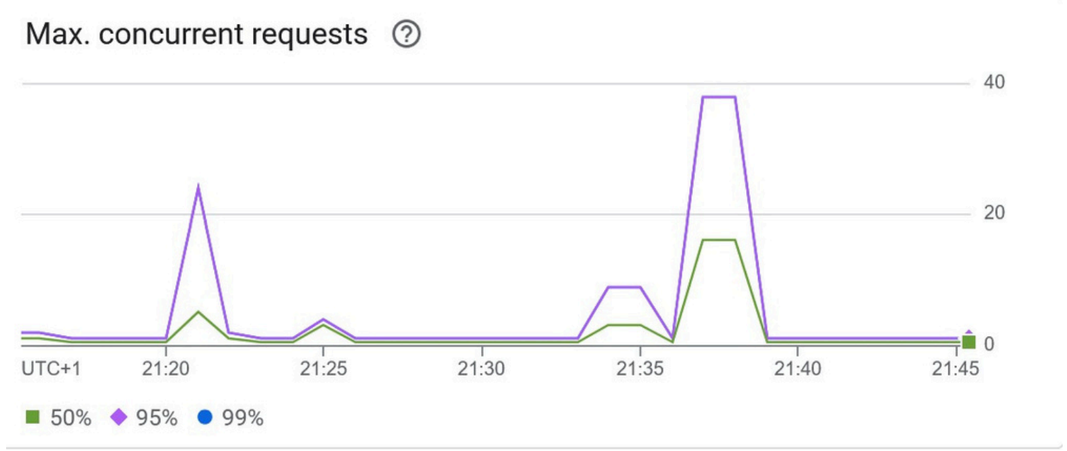
JMeter Statistics

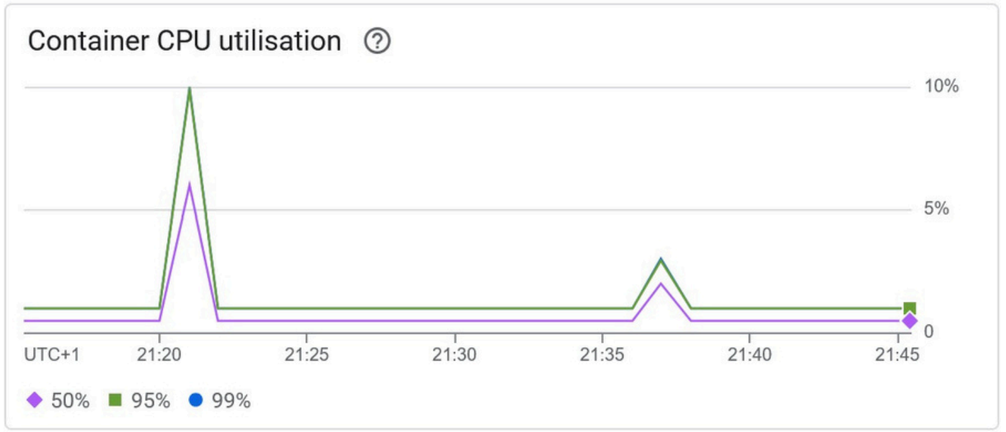
Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	50	0	0.00%	59678.82	47362	76028	56678.00	75468.30	75753.85	76028.00	0.63	0.28	0.31
POST / api/ predict/ map	50	0	0.00%	59678.82	47362	76028	56678.00	75468.30	75753.85	76028.00	0.63	0.28	0.31

Server-Side Analysis:
User Service (Gateway)

Observation: The User Service acted as an efficient, non-blocking gateway, successfully handling all incoming requests before dispatching them to the Prediction Service. Resource utilization remained low, indicating it was not a system bottleneck.

Cloud Run Metrics: User Service

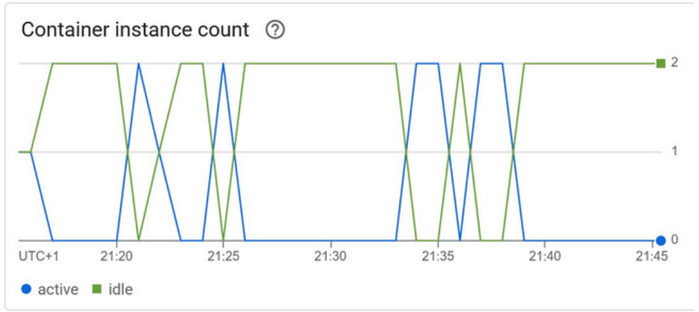


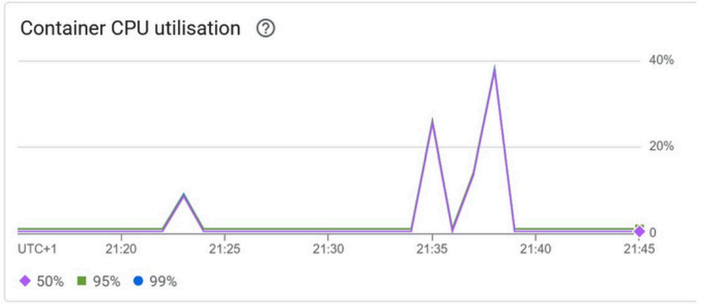


Server-Side Analysis: Prediction Service (Worker)

Observation: The Prediction Service demonstrated successful horizontal auto-scaling. It automatically provisioned new container instances in response to the increased request volume, distributing the computational load.

Cloud Run Metrics: Prediction Service



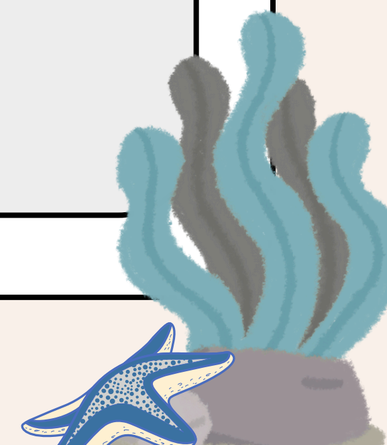


LET'S CHECK

Review

Achieved Goal

- Built accessible web-based HAB detection and prediction system
- Achieved shift: reactive → proactive HAB management
- 8-day predictions (90% accuracy) vs traditional 3-day (60% accuracy)
- Tiered architecture balances performance with cost-effectiveness
- Implemented tiered service model (free and premium).
- Potential to prevent millions in economic losses



Review

Remaining Problems

- Cold start delay after inactivity due to scale-to-zero setup.
- Prediction output is binary (toxic/non-toxic) without historical trends or confidence intervals
- YOLO detects green shades, sometimes confusing trees for algae.
- Static API keys instead of secure rotating tokens

Review

Improvements

- Add methodology explanations and data transparency (user feedback)
- Implement multi-threading optimization
- Expand YOLO dataset with higher-resolution, varied algae images
- Replace static API key with secure token-based authentication
- Integrate Application Performance Monitoring for debugging and optimization

Conclusions

Teamwork

- We split responsibilities early so everyone knew their focus.
- Daniel worked on data sources.
- Roshan and Dharmik handled the frontend and user experience.
- Kruthi and Dharmik worked on the machine learning models.
- Sagar and Karthika looked after backend and deployment.
- Sagar handled frontend-backend integration and integration testing.
- Regular Zoom calls and quick updates kept us in sync.
- If issues came up, we fixed them together without blocking progress.

Conclusions

Lessons Learned

- Clear communication and quick updates are essential.
- Testing early helps avoid last-minute issues.
- User feedback reveals gaps we might miss.
- Accuracy and speed must be balanced for real-world use.
- Cloud choices affect both cost and performance.
- Good documentation saves time later.

Conclusions

Improvements

- Gather more diverse, high-quality training data earlier.
- Start user testing sooner to refine the interface.
- Plan scalability from day one.
- Use better monitoring tools to track performance.
- Leave more time for tier integration.
- Automate deployment to save time.

Thank You